CLAIMS

What is claimed is:

1. A method of preventing buffer overrun security vulnerabilities comprising:

executing a modified call routine for placing a random amount of empty space onto a stack;

executing a called function; and

executing a modified return routine for removing said random amount of empty space from the stack.

2. The method of claim 1, wherein said modified call routine comprises:

placing a return address for the called function on the stack;

calculating a random number;

saving said random number in a secure location;

placing a plurality of blank bytes equal to the random number onto the stack;

building a stack frame by placing values from the called function onto the stack; and

setting an end of stack pointer to an end of the stack frame.

3. The method of claim 2, wherein said location is a processor register that is not generally accessible.

4. The method of claim 1, wherein said modified return routine comprises:

recalling a random number saved during an execution of said modified call routine;

removing a number of bytes equal to said random number from the stack;

retrieving a return address for the called function from the stack; and setting an end of stack pointer to an end of a previous stack frame.

5. The method of claim 1, wherein said modified call routine comprises:

placing a return address for the called function on the stack;

calculating a hash value of stack invariants;

saving said hash value in a secure location; and

building a stack frame by placing values from the called function onto the stack.

6. The method of claim 5, wherein said secure location is a processor register that is not generally accessible.

7. The method of claim 1, wherein said modified return routine comprises:

calculating a second hash value of stack invariants;

determining whether said second hash value matches a first hash value calculated during an execution of said modified call routine;

executing a stack corruption exception if said second hash value does not match said first hash value; and

setting an end of stack pointer to an end of a previous stack frame if said second hash value matches said first hash value.

8. A method of preventing buffer overrun security vulnerabilities comprising:

searching an executable program for all function calls at the time the executable is installed;

adding a random amount of blank space to all stacks generated by said function calls;

adjusting all references to said stacks to compensate for said blank space.

9. The method of claim 8, wherein said method is performed when said executable is installed.

10. The method of claim 9, further comprising saving said executable.

11. The method of claim 8, wherein said method is performed when said executable is loaded.

12. An apparatus comprising:

a storage device having stored therein one or more routines for preventing buffer overrun security vulnerabilities; and

a processor coupled to the storage device for executing the one or more routines that, when executing the routines, prevents buffer overrun errors by:

executing a modified call routine for placing a random amount of empty space onto a stack;

executing a called function; and

executing a modified return routine for removing said random amount of empty space from the stack.

13. The apparatus of claim 12, wherein said modified call routine comprises:

placing a return address for the called function on the stack;

calculating a random number;

saving said random number in a secure location;

placing a plurality of blank bytes equal to the random number onto the stack;

building a stack frame by placing values from the called function onto the stack; and

setting an end of stack pointer to an end of the stack frame.

14. The apparatus of claim 13, wherein said location is a processor register that is not generally accessible.

15. The apparatus of claim 12, wherein said modified return routine comprises:

recalling a random number saved during an execution of said modified call routine;

removing a number of bytes equal to said random number from the stack;

retrieving a return address for the called function from the stack; and

setting an end of stack pointer to an end of a previous stack frame.

16. The apparatus of claim 12, wherein said modified call routine comprises:

placing a return address for the called function on the stack;

calculating a hash value of stack invariants;

saving said hash value in a secure location; and

building a stack frame by placing values from the called function onto the stack.

17. The apparatus of claim 16, wherein said secure location is a processor register that is not generally accessible.

18. The apparatus of claim 12, wherein said modified return routine comprises:

calculating a second hash value of stack invariants;

determining whether said second hash value matches a first hash value calculated during an execution of said modified call routine;

executing a stack corruption exception if said second hash value does not match said first hash value; and setting an end of stack pointer to an end of a previous stack frame if said second hash value matches said first hash value.

19. An apparatus comprising:

a storage device having stored therein one or more routines for preventing buffer overrun security vulnerabilities; and

a processor coupled to the storage device for executing the one or more routines that, when executing the routines, prevents buffer overrun errors by:

searching an executable program for all function calls at the time the executable is installed;

adding a random amount of blank space to all stacks generated by said function calls;

adjusting all references to said stacks to compensate for said blank space.

20. The apparatus of claim 19, wherein said method is performed when said executable is installed.

21. The apparatus of claim 20, further comprising saving said executable.

22. The apparatus of claim 19, wherein said method is performed when said executable is loaded.

23. A machine-readable medium having stored thereon data representing sequences of instructions, said sequences of instructions which, when executed by a processor, cause said processor to prevents buffer overrun errors by:

executing a modified call routine for placing a random amount of empty space onto a stack;

executing a called function; and

executing a modified return routine for removing said random amount of empty space from the stack.

24. The machine-readable medium of claim 23, wherein said modified call routine comprises:
placing a return address for the called function on the stack;
calculating a random number;
saving said random number in a secure location;
placing a plurality of blank bytes equal to the random number onto the stack;
building a stack frame by placing values from the called function onto the stack; and
setting an end of stack pointer to an end of the stack frame.

25. The machine-readable medium of claim 24, wherein said location is a processor register that is not generally accessible.

26. The machine-readable medium of claim 23, wherein said modified return routine comprises:
recalling a random number saved during an execution of said modified call routine;
removing a number of bytes equal to said random number from the stack;
retrieving a return address for the called function from the stack; and
setting an end of stack pointer to an end of a previous stack frame.

27. The machine-readable medium of claim 23, wherein said modified call routine comprises:
placing a return address for the called function on the stack;
calculating a hash value of stack invariants;
saving said hash value in a secure location; and
building a stack frame by placing values from the called function onto the stack.

28. The machine-readable medium of claim 27, wherein said secure location is a processor register that is not generally accessible.

29. The machine-readable medium of claim 23, wherein said modified return routine comprises:
calculating a second hash value of stack invariants;
determining whether said second hash value matches a first hash value calculated during an execution of said modified call routine;
executing a stack corruption exception if said second hash value does not match said first hash value; and
setting an end of stack pointer to an end of a previous stack frame if said second hash value matches said first hash value.

30. A machine-readable medium having stored thereon data representing sequences of instructions, said sequences of instructions which, when executed by a processor, cause said processor to prevents buffer overrun errors by:
searching an executable program for all function calls at the time the executable is installed;
adding a random amount of blank space to all stacks generated by said function calls;
adjusting all references to said stacks to compensate for said blank space.

31. The machine-readable medium of claim 30, wherein said method is performed when said executable is installed.

32. The machine-readable medium of claim 31, further comprising saving said executable.

33. The machine-readable medium of claim 30, wherein said method is performed when said executable is loaded.